**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**A red and black logo

AI-generated content may be incorrect.**

**BÁO CÁO PROJECT**

**ĐỀ TÀI: THIẾT KẾ CƠ SỞ DỮ LIỆU**

**TRANG WEB BÁN GIÀY**

**MÔN HỌC: IT3290 – THỰC HÀNH CƠ SỞ DỮ LIỆU**

**GVHD: TS. NGUYỄN THỊ OANH; TS. TRẦN VĂN ĐẶNG**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nhóm** | **6** | | |
| **Họ và tên** | **MSSV** | **Mã lớp** | **Công việc** |
| Nguyễn Tài Hưng *(Leader)* | 20236034 | 156784 | Thiết kế database, vẽ ERD, vẽ RSD, viết trigger, function, view, truy vấn, làm báo cáo, thuyết trình. |
| Trần Việt Gia Khánh | 20235756 | 156784 | Viết trigger, function, truy vấn, làm báo cáo. |
| Nguyễn Trung Kiên | 20235759 | 156784 | Viết truy vấn, làm báo cáo. |

1. **MÔ TẢ NGHIỆP VỤ**
2. Đặt vấn đề

Trong bối cảnh thị trường thương mại điện tử ngày càng phát triển mạnh mẽ, đặc biệt là lĩnh vực thời trang và phụ kiện, việc sở hữu một trang web bán giày trực tuyến hiệu quả đã trở thành một yếu tố then chốt để các doanh nghiệp tiếp cận khách hàng tiềm năng và gia tăng doanh số. Tuy nhiên, sự đa dạng về mẫu mã, kích cỡ, nhà cung cấp và thông tin khách hàng đặt ra một thách thức không nhỏ trong việc quản lý dữ liệu một cách khoa học và hệ thống. Chính vì lẽ đó, dự án "Xây dựng cơ sở dữ liệu cho trang web bán giày" được đề xuất nhằm giải quyết bài toán này, tạo nền tảng vững chắc cho việc vận hành và phát triển trang web một cách bền vững.

1. Mô tả nghiệp vụ
2. Quản lí sản phẩm

* Doanh nghiệp nhập giày từ các Thương hiệu và phân loại chúng theo Danh mục.
* Mỗi sản phẩm có thể có nhiều Biến thể khác nhau về kích cỡ và màu sắc.
* Nhân viên có thể chọn thêm mới hoặc cập nhật sản phẩm để thêm sản phẩm mới, cập nhật thông tin, giả cả, mô tả,…
* Hệ thống theo dõi số lượng tồn kho của từng biến thể sản phẩm để đảm bảo chỉ bán các mặt hàng có sẵn.

1. Quản lý khách hàng

* Thêm/sửa/xóa thông tin khách hàng.
* Hỗ trợ cấp/đổi lại mật khẩu/thông tin khách hàng nếu cần thiết.

1. Quản lý doanh thu, lợi nhuận

* Hệ thống cần quản lý tổng doanh thu đơn hàng (chưa áp voucher, đã app voucher) thống kê theo từng ngày, tuần, tháng, quý, năm.
* Hệ thống quản lý số tiền nhận được của mỗi đơn hàng, cập theo thời gian thực.

1. Quá trình Bán hàng:

* Khách hàng tiềm năng tìm kiếm và duyệt xem các sản phẩm trên trang web.
* Khách hàng lựa chọn Biến thể sản phẩm (kích cỡ, màu sắc) và thêm vào giỏ hàng.
* Khách hàng tiến hành Đặt hàng, cung cấp thông tin giao hàng và thanh toán.
* Hệ thống tạo một Đơn hàng, ghi lại các Chi tiết đơn hàng cụ thể (sản phẩm, số lượng, giá tại thời điểm đặt) và trừ số lượng tồn kho tương ứng.
* Khách hàng thực hiện Thanh toán cho đơn hàng thông qua các phương thức được hỗ trợ.

1. Xử lý và Thực hiện Đơn hàng:

* Nhân viên quản lý đơn hàng tiếp nhận các Đơn hàng mới.
* Nhân viên xác nhận tính hợp lệ của đơn hàng và thông tin thanh toán.
* Đơn hàng được chuẩn bị (nhặt hàng từ kho, đóng gói).
* Quá trình Vận chuyển được khởi tạo, bàn giao gói hàng cho đơn vị vận chuyển và cập nhật mã theo dõi.
* Trạng thái Đơn hàng được cập nhật qua các giai đoạn (đang xử lý, đã đóng gói, đang vận chuyển, đã giao...).

1. **MÔ TẢ ỨNG DỤNG**
2. Kịch bản sử dụng

* Khách hàng: Khách hàng tìm kiếm, lựa chọn sản phẩm của cửa hàng, xác nhận đặt mua, thanh toán, viết đánh giá, liên hệ hỗ trợ, theo dõi tình trạng đơn hàng.
* Nhân viên: Quản lý thông tin khách hàng, quản lý đơn hàng, tạo đơn vận chuyển, hỗ trợ khách hàng.
* Quản lý: Quản lý thông tin khách hàng, quản lý voucher, quản lý kho, quản lý sản phẩm, thống kê doanh thu, lợi nhuận.

1. Mô tả chức năng
2. Khách hàng

* **Tìm kiếm và lọc sản phẩm:** Tìm kiếm sản phẩm theo từ khóa. Lọc sản phẩm theo các tiêu chí (giá, kích cỡ, màu sắc, thương hiệu, loại). Sắp xếp sản phẩm theo các tiêu chí (giá tăng/giảm, mới nhất, bán chạy).
* **Xem sản phẩm:** Xem chi tiết sản phẩm (tên, mã, thương hiệu, loại, kích cỡ, màu sắc, chất liệu, giá, mô tả, hình ảnh). Xem review, rating của sản phẩm. Xem các sản phẩm liên quan hoặc gợi ý.
* **Giỏ hàng:** Thêm sản phẩm vào giỏ hàng. Xem danh sách sản phẩm trong giỏ hàng. Thay đổi số lượng sản phẩm trong giỏ hàng. Xóa sản phẩm khỏi giỏ hàng. Cập nhật tổng tiền giỏ hàng.
* **Đặt hàng:** Nhập thông tin giao hàng (tên, địa chỉ, số điện thoại). Chọn phương thức thanh toán. Xem lại thông tin đơn hàng trước khi xác nhận. Xác nhận đặt hàng. Thanh toán (nếu là phương thức dùng thẻ).
* **Đánh giá sản phẩm:** Chọn vào danh sách các đơn hàng đã mua thành công. Chọn phần đánh giá. Viết đánh giá, thêm hình ảnh/video sản phẩm, đánh giá trên thang 5 điểm sản phẩm.
* **Quản lý tài khoản:** Đăng ký tài khoản mới. Đăng nhập/Đăng xuất. Xem và cập nhật thông tin cá nhân. Xem lịch sử đơn hàng và trạng thái đơn hàng. Hủy đơn hàng (trong thời gian cho phép).

1. Nhân viên

* **Quản lý sản phẩm (với quyền hạn):** Xem danh sách sản phẩm. Tìm kiếm và lọc sản phẩm. Xem chi tiết sản phẩm. Cập nhật thông tin sản phẩm (số lượng tồn kho, hình ảnh).
* **Quản lý đơn hàng:** Xem danh sách đơn hàng mới. Xem chi tiết đơn hàng. Xác nhận/ Từ chối đơn hàng. Cập nhật trạng thái đơn hàng (đã đặt hàng, đã xác nhận, đã gửi hàng, đang giao, đã giao). In hóa đơn.
* **Quản lý khách hàng:** Xem danh sách khách hàng. Xem thông tin chi tiết của khách hàng. Xem lịch sử mua hàng. Hỗ trợ khách hàng (VD: trả lời thắc mắc về đơn hàng).
* **Quản lý kho hàng (với quyền hạn):** Xem số lượng tồn kho.
* **Quảng lý tài khoản:** Đăng nhập/Đăng xuất. Xem và cập nhật thông tin cá nhân.

1. Quản lý

* **Quản lý sản phẩm:** Thêm sản phẩm mới. Cập nhật thông tin sản phẩm (toàn bộ thông tin). Xóa sản phẩm.
* **Quản lý kho hàng:** Xem số lượng tồn kho. Thêm sản phẩm mới. Điều chỉnh số lượng tồn kho. Xem báo cáo tồn kho.
* **Quản lý đơn hàng:** Xem toàn bộ danh sách đơn hàng. Xem chi tiết đơn hàng. Xác nhận đơn hàng. Hủy đơn hàng. Cập nhật trạng thái đơn hàng.
* **Quản lý người dùng:** Thêm tài khoản nhân viên mới. Sửa đổi thông tin tài khoản nhân viên. Xóa tài khoản nhân viên. Phân quyền truy cập cho nhân viên.
* **Báo cáo và thống kê:** Xem báo cáo doanh số (theo thời gian, sản phẩm, khách hàng, …). Xem thống kê sản phẩm bán chạy, sản phẩm tồn kho. Xuất dữ liệu báo cáo.

1. **SƠ ĐỒ THỰC THỂ LIÊN KẾT**
2. Xác định các thực thể chính

* Sản phẩm
* Biến thể của sản phẩm
* Phân loại
* Nhãn hàng
* Đơn hàng
* Chi tiết đơn hàng
* Khách hàng
* Nhân viên

1. Sơ đồ

A diagram of a diagram

AI-generated content may be incorrect.

(đính kèm file png)

1. **SƠ ĐỒ QUAN HỆ**
2. Xác định mối quan hệ

|  |  |  |  |
| --- | --- | --- | --- |
| Thực thể A | Thực thể B | Quan hệ | Mô tả liên kết |
| Customer | Order | 1 – n | Một khách hàng có thể đặt nhiều đơn hàng, mỗi đơn hàng chỉ được đặt bởi một khách hàng. |
| Order | Orderdetail | 1 – n | Mỗi đơn hàng có thể chứa nhiều chi tiết đơn hàng, mỗi chi tiết đơn hàng chỉ thuộc về một đơn hàng. |
| Employee | Order | n – n | Mỗi nhân viên có thể xử lý nhiều đơn hàng, mỗi đơn hàng có thể được xử lý bởi nhiều nhân viên. |
| Variant | Orderdetail | 1 – n | Mỗi biến thể có thể thuộc nhiều chi tiết đơn hàng, mỗi chi tiết đơn hàng chỉ có một biến thể. |
| Product | Variant | 1 – n | Một sản phẩm có thể có nhiều tùy chọn khác nhau (biến thể khác nhau), mỗi biến thể chỉ thuộc (là) một sản phẩm. |
| Category | Product | n – n | Mỗi danh mục có thể có nhiều sản phẩm, mỗi sản phẩm có thể thuộc nhiều danh mục. |
| Brand | Product | 1 – n | Một nhãn hàng có thể có nhiều sản phẩm, mỗi sản phẩm chỉ thuộc một nhãn hàng. |
| Feedback | Orderdetail | 1 – 1  hoặc  0 – 1 | Một chi tiết đơn hàng chỉ tương ứng duy nhất một đánh giá và ngược lại, hoặc không được đánh giá. |

1. Xác định các bảng

* category(**category\_id**, category\_name)
* brand(**brand**, brand\_name, country\_of\_origin, brand\_description)
* product(**product\_id**, *brand\_id*, product\_name, purchase\_price, selling\_price, material, product\_description, average\_rating)
* product\_category(***product\_id***, ***category\_id***) (product\_id, category\_id là **FK)**
* variant(**variant\_id**, *product\_id*, color, size, stock\_quantity)
* customer(**customer\_id**, first\_name, last\_name, street, city, district, phone\_number, email, credit\_card, dob, gender, username, password)
* feedback(***orderdetail\_id***, feedback, rating, feedback\_date) (orderdetail\_id là **FK**)
* orderdetail(**orderdetail\_id**, *order\_id*, *variant\_id*, order\_quantity, unit\_price, discount, sub\_total)
* order(**order\_id**, *customer\_id*, total\_amount, total\_discount, final\_amount, order\_date, status, payment\_method, note)
* employee(**employee\_id**, first\_name, last\_name, dob, phone\_number, email, username, password)
* status\_history(***employee\_id***, ***order\_id***, **date\_changed**, status) (employee\_id, order\_id là **FK**)

1. Sơ đồ

A screenshot of a computer

AI-generated content may be incorrect.

1. **CREATE TABLE**

CREATE TABLE customer (

    customer\_id CHAR(8) PRIMARY KEY,

    first\_name VARCHAR(30) NOT NULL,

    last\_name VARCHAR(30),

    street VARCHAR(50),

    city VARCHAR(30),

    district VARCHAR(30),

    phone\_number VARCHAR(10) UNIQUE NOT NULL,

    email VARCHAR(40) UNIQUE,

    credit\_card VARCHAR(30),

    dob DATE,

    gender CHAR(1) CHECK (gender IN ('F', 'M')),

    username VARCHAR(30) UNIQUE NOT NULL,

    password VARCHAR(20) NOT NULL

);

CREATE TABLE employee (

    employee\_id CHAR(8) PRIMARY KEY,

    first\_name VARCHAR(20),

    last\_name VARCHAR(20),

    dob DATE,

    phone\_number CHAR(10),

    email VARCHAR(40),

    username VARCHAR(30) UNIQUE NOT NULL,

    password VARCHAR(20) NOT NULL

);

CREATE TABLE category (

    category\_id CHAR(4) PRIMARY KEY,

    category\_name VARCHAR(30) NOT NULL

);

CREATE TABLE brand (

    brand\_id CHAR(4) PRIMARY KEY,

    brand\_name VARCHAR(100) NOT NULL,

    country\_of\_origin VARCHAR(40),

    brand\_description TEXT

);

CREATE TABLE product (

    product\_id CHAR(8) PRIMARY KEY,

    brand\_id CHAR(4),

    product\_name VARCHAR(200) NOT NULL,

    purchase\_price DECIMAL(10,2) NOT NULL,

    selling\_price DECIMAL(10,2) NOT NULL,

    material VARCHAR(50),

    product\_description TEXT,

    average\_rating DECIMAL(2,1) DEFAULT 0.0,

    CONSTRAINT fk\_brand\_id FOREIGN KEY (brand\_id) REFERENCES brand(brand\_id)

);

CREATE TABLE variant (

    variant\_id CHAR(8) PRIMARY KEY,

    product\_id CHAR(8) NOT NULL,

    color VARCHAR(50) NOT NULL,

    size DECIMAL(3,1) NOT NULL,

    stock\_quantity INTEGER NOT NULL DEFAULT 0 CHECK (stock\_quantity >= 0),

    CONSTRAINT fk\_product\_id FOREIGN KEY (product\_id) REFERENCES product(product\_id)

);

CREATE TABLE product\_category (

    product\_id CHAR(8) NOT NULL,

    category\_id CHAR(4) NOT NULL,

    CONSTRAINT pk\_product\_category PRIMARY KEY (product\_id, category\_id),

    CONSTRAINT fk\_product\_id FOREIGN KEY (product\_id) REFERENCES product(product\_id),

    CONSTRAINT fk\_category\_id FOREIGN KEY (category\_id) REFERENCES category(category\_id)

);

CREATE TABLE "order" (

    order\_id CHAR(8) PRIMARY KEY,

    customer\_id CHAR(8),

    total\_amount DECIMAL(10,2) DEFAULT 0.0,

    total\_discount DECIMAL(10,2) DEFAULT 0.0,

    final\_amount DECIMAL(10,2) DEFAULT 0.0,

    order\_date DATE DEFAULT CURRENT\_DATE,

    status VARCHAR(20) CHECK

        (status IN ('PENDING', 'PACKAGING', 'ON DELIVERY', 'DELIVERED', 'CANCELLED')) DEFAULT 'PENDING',

    last\_updated\_by\_employee\_id CHAR(8),

    payment\_method VARCHAR(20),

    note TEXT,

    CONSTRAINT fk\_customer\_id FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id),

    CONSTRAINT fk\_employee\_id FOREIGN KEY (last\_updated\_by\_employee\_id) REFERENCES employee(employee\_id)

);

CREATE TABLE orderdetail (

    orderdetail\_id CHAR(8) PRIMARY KEY,

    order\_id CHAR(8) NOT NULL,

    variant\_id CHAR(8) NOT NULL,

    order\_quantity INTEGER NOT NULL,

    unit\_price DECIMAL(10,2) DEFAULT 0.0,

    discount INTEGER DEFAULT 0,

    sub\_total DECIMAL(10,2) DEFAULT 0.0,

    CONSTRAINT fk\_order\_id FOREIGN KEY (order\_id) REFERENCES "order"(order\_id),

    CONSTRAINT fk\_variant\_id FOREIGN KEY (variant\_id) REFERENCES variant(variant\_id)

);

CREATE TABLE feedback (

    orderdetail\_id CHAR(8) PRIMARY KEY,

    feedback TEXT,

    rating DECIMAL(1,0),

    feedback\_date DATE,

    CONSTRAINT fk\_orderdetail\_id FOREIGN KEY (orderdetail\_id) REFERENCES orderdetail(orderdetail\_id)

);

CREATE TABLE status\_history (

    employee\_id CHAR(8) NOT NULL,

    order\_id CHAR(8) NOT NULL,

    date\_changed DATE NOT NULL DEFAULT CURRENT\_DATE,

    status VARCHAR(20) CHECK

        (status IN ('PENDING', 'PACKAGING', 'ON DELIVERY', 'DELIVERED', 'CANCELLED')) DEFAULT 'PENDING',

    CONSTRAINT pk\_status\_history PRIMARY KEY (employee\_id, order\_id, date\_changed),

    CONSTRAINT fk\_employee\_id FOREIGN KEY (employee\_id) REFERENCES employee(employee\_id),

    CONSTRAINT fk\_order\_id FOREIGN KEY (order\_id) REFERENCES "order"(order\_id)

);

1. **VIEW**

-- Tạo view chỉ hiện thị các các sản phẩm còn hàng

CREATE OR REPLACE VIEW available\_products AS

SELECT p.\*

FROM product p

WHERE

EXISTS (

SELECT 1

FROM variant v

WHERE v.product\_id = p.product\_id AND v.stock\_quantity > 0

);

1. **INDEX**

CREATE INDEX idx\_order\_status\_date ON "order"(status, order\_date ASC);

CREATE INDEX idx\_order\_customer\_date ON "order"(customer\_id, order\_date DESC);

CREATE INDEX idx\_variant\_product\_stock ON variant(product\_id, stock\_quantity);

CREATE INDEX idx\_variant\_size ON variant(size);

1. **FUNCTION**

**-- Hàm check stock của một variant còn bao nhiêu**

CREATE OR REPLACE FUNCTION fn\_check\_stock(p\_variant\_id CHAR(8))

RETURNS INTEGER AS $$

DECLARE

v\_stock\_quantity INTEGER;

BEGIN

SELECT

stock\_quantity

INTO

v\_stock\_quantity

FROM

variant

WHERE

variant\_id = p\_variant\_id;

RETURN COALESCE(v\_stock\_quantity, 0);

END;

$$ LANGUAGE plpgsql;

**-- Hàm tìm kiếm sản phẩm dựa trên một chuỗi đầu vào**

CREATE OR REPLACE FUNCTION fn\_search\_products(p\_search\_term TEXT)

RETURNS TABLE(product\_id CHAR(8)) AS $$

BEGIN

RETURN QUERY

SELECT DISTINCT

p.product\_id

FROM

product p

JOIN

brand b ON p.brand\_id = b.brand\_id

LEFT JOIN

variant v ON p.product\_id = v.product\_id

WHERE

p.product\_name ILIKE '%' || p\_search\_term || '%' OR

b.brand\_name ILIKE '%' || p\_search\_term || '%' OR

p.product\_description ILIKE '%' || p\_search\_term || '%' OR

v.color ILIKE '%' || p\_search\_term || '%';

END;

$$ LANGUAGE plpgsql;

**-- Hàm tạo order, đóng gói toàn bộ vào một giao dịch**

CREATE OR REPLACE FUNCTION fn\_create\_order(

p\_customer\_id CHAR(8),

p\_payment\_method VARCHAR(20),

p\_note TEXT,

p\_cart\_items JSONB -- '[{"variant\_id": "V0000001", "quantity": 2}, ...]'

)

RETURNS CHAR(8) AS $$

DECLARE

v\_new\_order\_id CHAR(8);

v\_item JSONB;

v\_current\_stock INTEGER;

v\_product\_price DECIMAL(10, 2);

v\_total\_amount DECIMAL(10, 2) := 0;

v\_final\_amount DECIMAL(10, 2) := 0;

BEGIN

FOR v\_item IN SELECT \* FROM jsonb\_array\_elements(p\_cart\_items)

LOOP

v\_current\_stock := fn\_check\_stock((v\_item->>'variant\_id')::CHAR(8));

IF v\_current\_stock < (v\_item->>'quantity')::INTEGER THEN

RAISE EXCEPTION 'Insufficient inventory with variant %. The remaining quantity is %', v\_item->>'variant\_id', v\_current\_stock;

END IF;

END LOOP;

INSERT INTO "order" (customer\_id, payment\_method, note, status, order\_date)

VALUES (p\_customer\_id, p\_payment\_method, p\_note, 'PENDING', NOW())

RETURNING order\_id INTO v\_new\_order\_id;

FOR v\_item IN SELECT \* FROM jsonb\_array\_elements(p\_cart\_items)

LOOP

SELECT p.selling\_price INTO v\_product\_price

FROM variant v

JOIN product p ON v.product\_id = p.product\_id

WHERE v.variant\_id = (v\_item->>'variant\_id')::CHAR(8);

INSERT INTO orderdetail (order\_id, variant\_id, order\_quantity, unit\_price, discount, sub\_total)

VALUES (

v\_new\_order\_id,

(v\_item->>'variant\_id')::CHAR(8),

(v\_item->>'quantity')::INTEGER,

v\_product\_price,

0,

v\_product\_price \* (v\_item->>'quantity')::INTEGER

);

UPDATE variant

SET stock\_quantity = stock\_quantity - (v\_item->>'quantity')::INTEGER

WHERE variant\_id = (v\_item->>'variant\_id')::CHAR(8);

END LOOP;

SELECT

SUM(unit\_price \* order\_quantity),

SUM(sub\_total)

INTO

v\_total\_amount, v\_final\_amount

FROM orderdetail

WHERE order\_id = v\_new\_order\_id;

UPDATE "order"

SET

total\_amount = v\_total\_amount,

final\_amount = v\_final\_amount

WHERE order\_id = v\_new\_order\_id;

RETURN v\_new\_order\_id;

END;

$$ LANGUAGE plpgsql;

-- Hàm tính tổng doanh thu của một thương hiệu

CREATE OR REPLACE FUNCTION brand\_revenue(bid CHAR(4))

RETURNS NUMERIC AS $$

DECLARE

revenue NUMERIC;

BEGIN

SELECT SUM(od.sub\_total) INTO revenue

FROM orderdetail od

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

WHERE p.brand\_id = bid;

RETURN COALESCE(revenue, 0);

END;

$$ LANGUAGE plpgsql;

1. **TRIGGER**

**-- Hàm cho trigger tự động lưu lịch sử thay đổi trạng thái đơn hàng**

CREATE OR REPLACE FUNCTION fn\_handle\_after\_update\_order()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.status IS DISTINCT FROM OLD.status THEN

INSERT INTO status\_history (order\_id, employee\_id, date\_changed, status)

VALUES (NEW.order\_id, NEW.last\_update\_by\_employee\_id, NOW(), NEW.status);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_after\_update\_order

AFTER UPDATE ON "order"

FOR EACH ROW

EXECUTE FUNCTION fn\_handle\_after\_update\_order();

**-- Hàm cho trigger khi có một đơn hàng bị cập nhật trạng thái thành CANCELLED**

CREATE OR REPLACE FUNCTION fn\_handle\_order\_cancellation()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.status = 'CANCELLED' AND OLD.status <> 'CANCELLED' THEN

UPDATE variant v

SET stock\_quantity = v.stock\_quantity + od.order\_quantity

FROM orderdetail od

WHERE v.variant\_id = od.variant\_id AND od.order\_id = NEW.order\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_handle\_order\_cancellation

AFTER UPDATE ON "order"

FOR EACH ROW

EXECUTE FUNCTION fn\_handle\_order\_cancellation();

**-- Hàm cho trigger sau khi có feedback mới**

CREATE OR REPLACE FUNCTION fn\_handle\_after\_insert\_feedback()

RETURNS TRIGGER AS $$

DECLARE

v\_product\_id CHAR(8);

v\_new\_avg\_rating DECIMAL(2,1);

BEGIN

SELECT

v.product\_id INTO v\_product\_id

FROM

orderdetail od

JOIN

variant v ON od.variant\_id = v.variant\_id

WHERE

od.orderdetail\_id = NEW.orderdetail\_id;

SELECT

AVG(f.rating) INTO v\_new\_avg\_rating

FROM

feedback f

JOIN

orderdetail od ON f.orderdetail\_id = od.orderdetail\_id

JOIN

variant v ON od.variant\_id = v.variant\_id

WHERE

v.product\_id = v\_product\_id;

UPDATE product

SET average\_rating = v\_new\_avg\_rating

WHERE product\_id = v\_product\_id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_after\_insert\_feedback

AFTER INSERT ON feedback

FOR EACH ROW

EXECUTE FUNCTION fn\_handle\_after\_insert\_feedback();

**--Trigger tự động tính toán và thiết lập giá trị của unit\_price và sub\_total của orderdetail**

CREATE OR REPLACE FUNCTION calculate\_orderdetail\_prices\_trigger\_function()

RETURNS TRIGGER AS $$ DECLARE v\_selling\_price NUMERIC(10, 2);

BEGIN

SELECT p.selling\_price INTO v\_selling\_price

FROM product AS p JOIN variant AS v ON p.product\_id = v.product\_id

WHERE v.variant\_id = NEW.variant\_id;

IF NOT FOUND THEN RAISE EXCEPTION 'Product variant with ID % not found.', NEW.variant\_id;  
END IF;  
NEW.unit\_price = v\_selling\_price;  
NEW.sub\_total = NEW.unit\_price \* NEW.order\_quantity \* (1 - NEW.discount / 100.00);  
RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_set\_orderdetail\_prices

BEFORE INSERT ON orderdetail FOR EACH ROW EXECUTE FUNCTION calculate\_orderdetail\_prices\_trigger\_function();

**-- Trigger tự động cập nhật bảng order khi orderline được cập nhật**

CREATE OR REPLACE FUNCTION update\_order\_totals\_function()

RETURNS TRIGGER AS $$

DECLARE

v\_order\_id CHAR(8);

v\_total\_amount NUMERIC(10, 2);

v\_total\_discount\_calculated NUMERIC(10, 2);

v\_final\_amount\_calculated NUMERIC(10, 2);

BEGIN

IF TG\_OP = 'DELETE' THEN v\_order\_id = OLD.order\_id;

ELSE v\_order\_id = NEW.order\_id; END IF;

SELECT  
 SUM(od.order\_quantity \* od.unit\_price),  
 SUM(od.order\_quantity \* od.unit\_price \* od.discount / 100.00),  
 SUM(od.sub\_total)  
INTO  
 v\_total\_amount,  
 v\_total\_discount\_calculated,  
 v\_final\_amount\_calculated  
FROM  
 orderdetail od  
WHERE  
 od.order\_id = v\_order\_id;  
  
UPDATE "order"  
SET  
 total\_amount = v\_total\_amount,  
 total\_discount = v\_total\_discount\_calculated,  
 final\_amount = v\_final\_amount\_calculated  
WHERE  
 order\_id = v\_order\_id;  
  
IF TG\_OP = 'DELETE' THEN  
 RETURN OLD;  
ELSE  
 RETURN NEW;  
END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_after\_orderdetail\_change

AFTER INSERT OR UPDATE OR DELETE ON orderdetail

FOR EACH ROW

EXECUTE FUNCTION update\_order\_totals\_function();

**--Trigger quản lý số lượng tồn kho của sản phẩm trong variant**

CREATE OR REPLACE FUNCTION update\_stock\_quantity\_function()

RETURNS TRIGGER AS $$

DECLARE v\_stock\_quantity INT; v\_quantity\_change INT;

BEGIN

SELECT stock\_quantity INTO v\_stock\_quantity

FROM variant

WHERE variant\_id = COALESCE(NEW.variant\_id, OLD.variant\_id);

IF TG\_OP = 'INSERT' THEN  
 IF v\_stock\_quantity < NEW.order\_quantity THEN  
 RAISE EXCEPTION 'Không đủ số lượng tồn kho cho sản phẩm có variant\_id %. Chỉ còn % sản phẩm.', NEW.variant\_id, v\_stock\_quantity;  
 END IF;  
 UPDATE variant  
 SET stock\_quantity = stock\_quantity - NEW.order\_quantity  
 WHERE variant\_id = NEW.variant\_id;  
  
ELSIF TG\_OP = 'DELETE' THEN  
 UPDATE variant  
 SET stock\_quantity = stock\_quantity + OLD.order\_quantity  
 WHERE variant\_id = OLD.variant\_id;  
  
ELSIF TG\_OP = 'UPDATE' THEN  
 v\_quantity\_change = NEW.order\_quantity - OLD.order\_quantity;

IF v\_stock\_quantity < v\_quantity\_change THEN  
 RAISE EXCEPTION 'Không đủ số lượng tồn kho để tăng số lượng cho sản phẩm có variant\_id %. Chỉ còn % sản phẩm.', NEW.variant\_id, v\_stock\_quantity;  
 END IF;  
 UPDATE variant  
 SET stock\_quantity = stock\_quantity - v\_quantity\_change  
 WHERE variant\_id = NEW.variant\_id;  
END IF;  
IF TG\_OP = 'DELETE' THEN  
 RETURN OLD;  
ELSE  
 RETURN NEW;  
END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_before\_orderdetail\_change\_update\_stock

BEFORE INSERT OR UPDATE OR DELETE ON orderdetail

FOR EACH ROW

EXECUTE FUNCTION update\_stock\_quantity\_function();

1. **QUERY**
2. **Truy vấn nghiệp vụ quản lí sản phẩm và kho hàng**

-- Nguyễn Trung Kiên

-- 1. Liệt kê tất cả sản phẩm kèm tên thương hiệu và giá bán

SELECT p.product\_id, p.product\_name, b.brand\_name, p.selling\_price

FROM product p

JOIN brand b ON p.brand\_id = b.brand\_id

LIMIT 10;

-- 2. Liệt kê 10 sản phẩm có giá bán cao nhất (product\_id, product\_name, selling\_price)

SELECT product\_id, product\_name, selling\_price

FROM product

ORDER BY selling\_price DESC

LIMIT 10;

-- 3. Thống kê số sản phẩm theo từng thương hiệu

SELECT b.brand\_id, b.brand\_name, COUNT(p.product\_id) AS product\_count

FROM brand b

LEFT JOIN product p ON b.brand\_id = p.brand\_id

GROUP BY b.brand\_id, b.brand\_name

ORDER BY product\_count DESC;

-- 4. Liệt kê các sản phẩm chưa từng được bán (không xuất hiện trong orderdetail)

SELECT p.product\_id, p.product\_name

FROM product p

LEFT JOIN variant v ON p.product\_id = v.product\_id

LEFT JOIN orderdetail od ON v.variant\_id = od.variant\_id

WHERE od.orderdetail\_id IS NULL

LIMIT 10;

-- 5. Thống kê số lượng sản phẩm theo từng phân loại (category)

SELECT c.category\_id, c.category\_name, COUNT(pc.product\_id) AS product\_count

FROM category c

LEFT JOIN product\_category pc ON c.category\_id = pc.category\_id

GROUP BY c.category\_id, c.category\_name

ORDER BY product\_count DESC;

-- 6. Thống kê số lượng sản phẩm theo từng nước xuất xứ của thương hiệu

SELECT b.country\_of\_origin, COUNT(p.product\_id) AS product\_count

FROM brand b

JOIN product p ON b.brand\_id = p.brand\_id

GROUP BY b.country\_of\_origin

ORDER BY product\_count DESC;

-- 7. Liệt kê 3 thương hiệu có doanh thu cao nhất và số sản phẩm đã bán của mỗi thương hiệu

WITH BrandSales AS (

SELECT

b.brand\_id,

b.brand\_name,

SUM(od.sub\_total) AS total\_revenue,

SUM(od.order\_quantity) AS total\_sold

FROM brand b

JOIN product p ON b.brand\_id = p.brand\_id

JOIN variant v ON p.product\_id = v.product\_id

JOIN orderdetail od ON v.variant\_id = od.variant\_id

GROUP BY b.brand\_id, b.brand\_name

)

SELECT \*

FROM BrandSales

ORDER BY total\_revenue DESC

LIMIT 3;

-- 8. Liệt kê các thương hiệu có tổng số lượng tồn kho của tất cả sản phẩm lớn hơn 100

SELECT b.brand\_id, b.brand\_name, SUM(v.stock\_quantity) AS total\_stock

FROM brand b

JOIN product p ON b.brand\_id = p.brand\_id

JOIN variant v ON p.product\_id = v.product\_id

GROUP BY b.brand\_id, b.brand\_name

HAVING SUM(v.stock\_quantity) > 100;

-- 9. Sử dụng function: Lấy trung bình đánh giá của một sản phẩm

CREATE OR REPLACE FUNCTION get\_average\_rating(pid CHAR(8))

RETURNS NUMERIC(2,1) AS $$

DECLARE

avg\_rating NUMERIC(2,1);

BEGIN

SELECT average\_rating INTO avg\_rating FROM product WHERE product\_id = pid;

RETURN avg\_rating;

END;

$$ LANGUAGE plpgsql;

SELECT product\_id, product\_name, get\_average\_rating(product\_id) AS avg\_rating

FROM product

ORDER BY avg\_rating DESC

LIMIT 10;

-- 10. FUNCTION: Trả về tổng doanh thu của một thương hiệu theo brand\_id

SELECT brand\_id, brand\_name, brand\_revenue(brand\_id) AS total\_revenue

FROM brand

ORDER BY total\_revenue DESC

LIMIT 5;

1. **Truy vấn nghiệp vụ đặt hàng và quản lí đơn hàng**

-- Nguyễn Tài Hưng

-- 1. Tìm kiếm sản phẩm theo khoảng giá

SELECT \*

FROM available\_products

WHERE selling\_price

BETWEEN 40 AND 50

ORDER BY selling\_price ASC

LIMIT 10;

-- 2. Tìm kiếm sản phẩm theo size

SELECT DISTINCT p.\*

FROM available\_products p

JOIN variant v

USING (product\_id)

WHERE v.size = 44

ORDER BY p.selling\_price ASC

LIMIT 10;

-- 3. Lấy tất cả đánh giá của một sản phẩm

SELECT v.product\_id, v.variant\_id, v.color, f.rating, f.feedback, f.feedback\_date, CONCAT(c.first\_name, ' ', c.last\_name) customer\_name

FROM feedback f

JOIN orderdetail od

ON f.orderdetail\_id = od.orderdetail\_id

JOIN variant v

ON od.variant\_id = v.variant\_id

JOIN "order" o

ON od.order\_id = o.order\_id

JOIN customer c

ON o.customer\_id = c.customer\_id

WHERE v.product\_id = 'PD000020';

-- 4. Tìm kiếm các sản phẩm đang được bán theo một chuỗi

SELECT \* FROM fn\_search\_products('sport');

-- 5. Tạo một đơn hàng mới với một vài biến thể

SELECT fn\_create\_order(

'CUS00001',

'Cash',

'hehehe!',

'[{"variant\_id": "V0000020", "quantity": 5}, {"variant\_id": "V0000040", "quantity": 3}]'::JSONB

);

-- 6. Xem chi tiết một đơn hàng

SELECT o.\*, p.product\_name, v.color, v.size, od.order\_quantity, od.unit\_price

FROM "order" o

JOIN orderdetail od ON o.order\_id = od.order\_id

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

WHERE o.order\_id = 'ORD00308';

-- 7. Lấy lịch sử đơn hàng của một khách hàng cụ thể

SELECT order\_id, order\_date, final\_amount, status

FROM "order"

WHERE customer\_id = 'CUS00001' -- Thay bằng ID của khách hàng đang đăng nhập

ORDER BY order\_date DESC;

-- 8. Lấy danh sách các đơn hàng ở trạng thái "PENDING" để nhân viên vào xử lý

SELECT order\_id, customer\_id, final\_amount, order\_date, payment\_method

FROM "order"

WHERE status IN ('pending', 'processing')

ORDER BY order\_date ASC; -- Ưu tiên xử lý đơn cũ trước

--9. Tìm kiếm đơn hàng theo thông tin khách hàng (SĐT hoặc Tên)

SELECT o.order\_id, o.order\_date, o.status, o.final\_amount, c.first\_name, c.last\_name, c.phone\_number

FROM "order" o

JOIN customer c ON o.customer\_id = c.customer\_id

WHERE c.phone\_number = '0987654321' OR c.first\_name ILIKE '%Hung%';

--10. Lấy danh sách sản phẩm cần đóng gói cho các đơn đang xử lý

SELECT p.product\_id, p.product\_name, v.color, v.size, SUM(od.order\_quantity) AS total\_quantity\_to\_pack

FROM orderdetail od

JOIN "order" o ON od.order\_id = o.order\_id

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

WHERE o.status = 'PENDING'

GROUP BY p.product\_id, p.product\_name, v.color, v.size

ORDER BY p.product\_name;

--11. Xem lịch sử thay đổi trạng thái của một đơn hàng

SELECT s.date\_changed, s.status, e.first\_name || ' ' || e.last\_name AS employee\_name

FROM status\_history s

LEFT JOIN employee e ON s.employee\_id = e.employee\_id

WHERE s.order\_id = 'OD000001'

ORDER BY s.date\_changed ASC;

--12. Liệt kê các đơn hàng bị treo quá lâu

SELECT order\_id, customer\_id, order\_date, status

FROM "order"

WHERE status = 'PENDING' AND order\_date < NOW() - INTERVAL '3 days';

--13. Top 10 khách hàng chi tiêu nhiều nhất

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email, COUNT(o.order\_id) AS total\_orders, SUM(o.final\_amount) AS total\_spent

FROM customer c

JOIN "order" o ON c.customer\_id = o.customer\_id

WHERE o.status = 'completed'

GROUP BY c.customer\_id, c.first\_name, c.last\_name, c.email

ORDER BY total\_spent DESC

LIMIT 10;

--14. Tính tổng giá trị của hàng tồn kho hiện tại

SELECT SUM(v.stock\_quantity \* p.purchase\_price) AS total\_inventory\_value

FROM variant v

JOIN product p ON v.product\_id = p.product\_id

WHERE v.stock\_quantity > 0;

1. **Truy vấn về báo cáo và thống kê**

--Trần Việt Gia Khánh

-- 1. Báo cáo doanh thu hàng ngày/tháng/năm

-- Doanh thu theo ngày

SELECT

    order\_date,

    SUM(final\_amount) AS daily\_revenue

FROM "order"

WHERE order\_date = CURRENT\_DATE -- Hoặc thay bằng 'YYYY-MM-DD' cụ thể

GROUP BY order\_date

ORDER BY order\_date;

-- Doanh thu theo tháng

SELECT

    TO\_CHAR(order\_date, 'YYYY-MM') AS month,

    SUM(final\_amount) AS monthly\_revenue

FROM "order"

WHERE EXTRACT(YEAR FROM order\_date) = EXTRACT(YEAR FROM CURRENT\_DATE)

GROUP BY month

ORDER BY month;

-- Doanh thu theo năm

SELECT

    EXTRACT(YEAR FROM order\_date) AS year,

    SUM(final\_amount) AS yearly\_revenue

FROM "order"

GROUP BY year

ORDER BY year;

-- 2. Báo cáo sản phẩm bán chạy nhất (theo số lượng hoặc doanh thu)

-- Sản phẩm bán chạy nhất theo số lượng

SELECT p.product\_id, p.product\_name, SUM(od.order\_quantity) AS total\_quantity\_sold

FROM orderdetail od

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

GROUP BY p.product\_id, p.product\_name

ORDER BY total\_quantity\_sold DESC

LIMIT 10;

-- Sản phẩm bán chạy nhất theo doanh thu

SELECT p.product\_id, p.product\_name, SUM(od.sub\_total) AS total\_revenue

FROM orderdetail od

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

GROUP BY p.product\_id, p.product\_name ORDER BY total\_revenue DESC LIMIT 10;

-- 3. Báo cáo doanh thu theo thương hiệu

SELECT b.brand\_id, b.brand\_name, SUM(od.sub\_total) AS brand\_revenue

FROM orderdetail od

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

JOIN brand b ON p.brand\_id = b.brand\_id

GROUP BY b.brand\_id, b.brand\_name

ORDER BY brand\_revenue DESC;

-- 4. Báo cáo doanh thu theo danh mục sản phẩm

SELECT c.category\_id, c.category\_name, SUM(od.sub\_total) AS category\_revenue

FROM orderdetail od

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

JOIN product\_category pc ON p.product\_id = pc.product\_id

JOIN category c ON pc.category\_id = c.category\_id

GROUP BY c.category\_id, c.category\_name

ORDER BY category\_revenue DESC;

-- 5. Lịch sử mua hàng của một khách hàng cụ thể

SELECT o.order\_id, o.order\_date, p.product\_name, v.color, v.size, od.order\_quantity, od.unit\_price, od.sub\_total

FROM "order" o

JOIN orderdetail od ON o.order\_id = od.order\_id

JOIN variant v ON od.variant\_id = v.variant\_id

JOIN product p ON v.product\_id = p.product\_id

WHERE o.customer\_id = 'CUST001'

ORDER BY o.order\_date DESC, o.order\_id DESC;

-- 6. Danh sách khách hàng có số lượng đơn hàng cao nhất

SELECT

c.customer\_id,

    c.first\_name,

    c.last\_name,

    COUNT(o.order\_id) AS total\_orders

FROM customer c

JOIN "order" o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY total\_orders DESC

LIMIT 10;

-- 7. Danh sách khách hàng có tổng giá trị mua hàng cao nhất

SELECT

c.customer\_id,

    c.first\_name,

    c.last\_name,

    SUM(o.final\_amount) AS total\_spending

FROM customer c

JOIN "order" o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY total\_spending DESC

LIMIT 10;

-- 8. Các đơn hàng bị hủy hoặc hoàn trả

SELECT

    o.order\_id,

    o.order\_date,

    o.final\_amount,

    o.status,

    c.first\_name,

    c.last\_name

FROM "order" o

JOIN customer c ON o.customer\_id = c.customer\_id

WHERE o.status IN ('Cancelled', 'Returned', 'Refunded')

ORDER BY o.order\_date DESC;